Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1-69. (Canceled)

70. (Previously Presented) A processor comprising:

a plurality of execution units to execute a plurality of threads;

suspend logic to set a monitor address in response to a first instruction in a first thread according to an implicit operand in a predetermined register and to suspend the first thread in response to a second instruction of the first thread;

a monitor to cause resumption of the first thread in response to a memory access to the monitor address.

71. (Previously Presented) The processor of claim 70 wherein said monitor is to cause resumption of the first thread in response to events that cause a translation look-aside buffer to be flushed.

72. (Previously Presented) The processor of claim 70 wherein said monitor is to cause resumption of the first thread in response to a write to a control register CR0.

73. (Currently Amended) The processor of claim ~~72~~ 70 wherein said monitor is to cause resumption of the first thread in response to an interrupt being one of a non-maskable interrupt (NMI) and a system management interrupt (SMI) ~~or a fault~~.

74. (Currently Amended) The processor of claim 70 wherein said suspend logic is only to suspend said first thread if said monitor address is associated with a selected memory type.

75. (Currently Amended) The processor of claim ~~73~~ 70 wherein said suspend logic is only to suspend said first thread if said memory access ~~monitor address~~ is a write access to a selected memory ~~type~~.

76. (Previously Presented) The processor of claim 74 wherein said selected memory type is a write-back type of memory.

77. (Cancelled).

78. (Currently Amended) The processor of claim 77 wherein a powerdown event ~~is not a monitor break event~~does not cause resumption of the first thread.

79. (Currently Amended) The processor of claim 70 further comprising an instruction buffer which can be adapted as ~~combined to form~~ a single partition dedicated to one thread or ~~can be~~ partitioned to be used by the plurality of threads.

80. (Previously Presented) The processor of claim 70 further comprising a front end, which performs micro-operation (uOP) generation, generating uOPs from macroinstructions.

81. (Previously Presented) The processor of claim 80 wherein said processor is capable of out-of-order execution and wherein said first instruction is followed by a store fence.

82. (Previously Presented) The processor of claim 70 wherein a second operand specifies events to mask.

83. (Previously Presented) The processor of claim 82 wherein one mask bit indicates that masked interrupts break restart the first thread despite interrupts being masked.

84. (Previously Presented) The processor of claim 70 wherein said first instruction is an instruction having only implicit operands.

85. (Previously Presented) The processor of claim 70 further comprising:

coherency logic to perform a read line transaction in conjunction with suspending the first thread.

86. (Previously Presented) The processor of claim 85 wherein said coherency logic is to perform a cache line flush to flush internal caches in conjunction with suspending the first thread.

87. (Previously Presented) The processor of claim 70 wherein said processor is to monitor for a request for ownership or an invalidate cycle to the monitor address.

88. (Previously Presented) The processor of claim 70 wherein said processor is to assert a hit signal during a snoop phase of a bus transaction implicating the monitor address.

89. (Currently Amended) The processor of claim 88 wherein the monitor is to cause said plurality of partitionable resources to be re-partitioned to accommodate execution of ~~said~~ a first thread in response to the memory access to the monitor address.

90. (Previously Presented) The processor of claim 89 wherein said plurality of partitionable resources comprise:

an instruction queue;

a re-order buffer;

a pool of registers;

a plurality of store buffers.

91. (Previously Presented) The processor of claim 90 further comprising:

a plurality of duplicated resources, said plurality of duplicated resources being duplicated for each of said plurality of threads, said plurality of duplicated resources comprising:

a plurality of processor state variables;

an instruction pointer;

register renaming logic.

92. (Previously Presented) The processor of claim 91 further comprising:

a plurality of shared resources, said plurality of shared resources being available for use by any of said plurality of threads, said plurality of shared resources comprising:

said plurality of execution units;

a cache;

a scheduler.

93. (Previously Presented) A processor comprising:

a front end to receive a first instruction and a second instruction, the first instruction having an implicit operand from a predetermined register indicating a monitor address;

execution resources to execute the first instruction and the second instruction and to enter a first implementation dependent state in response to the second instruction if the first instruction has been executed and no break events have occurred after execution of the first instruction;

a monitor to cause exit from the first implementation dependent state in response to a memory access to the monitor address.

94. (Previously Presented) The processor of claim 93 wherein said implicit operand is to indicate a linear address, and wherein said processor further comprises address translation logic to translate said linear address to obtain the monitor address which is a physical address.

95. (Previously Presented) The processor of claim 93 further comprising:

coherency logic to ensure that no cache in another processor coupled to the processor stores information at said monitor address in a modified or exclusive state.

96. (Previously Presented) The processor of claim 93 wherein said coherency logic is to assert a hit signal in response to another processor snooping the monitor address.

97. (Previously Presented) The processor of claim 95 wherein said coherency logic is to assert a hit signal in response to another processor snooping the monitor address.

98. (Previously Presented) A method comprising:

receiving a first opcode executing in a first thread of execution;

translating a linear address associated with said first opcode into a physical address;

executing a bus transaction by a monitoring bus agent to ensure no other bus agent has sufficient ownership of data associated with said physical address to allow another bus agent to modify the data without informing the monitoring bus agent;

monitoring for an access to said physical address;

signaling a hit if another bus agent reads said physical address;

receiving a second opcode in the first thread of execution;

suspending said first thread of execution and enabling recognition of a monitor event in response to the second opcode;

resuming said first thread if the access occurs;

resuming execution of the first thread in response to any one of a first set of events.

99. (Currently Amended) The method of claim 98 further comprising:

detecting a second set of events; and

ignoring said~~a~~ second set of events differing from said first set of events.

100. (Previously Presented) The method of claim 98 wherein said access is a write access.

101. (Currently Amended) The method of claim 98 wherein said linear address ~~must~~ corresponds to an address of a predetermined type of memory as a precondition to said first thread being suspended.

102. (Previously Presented) The method of claim 101 wherein said predetermined type of memory is write back memory.

103. (Previously Presented) The method of claim 98 wherein said bus transaction is a read bus transaction.

104. (Previously Presented) The method of claim 98 further comprising:

relinquishing a plurality of partitioned resources in response to the first thread being suspended;

partitioning a plurality of resources in response to the access.

105. (Previously Presented) The method of claim 104 wherein said plurality of partitioned resources comprise:

an instruction queue;

a re-order buffer;

a pool of registers;

a plurality of store buffers.

106. (Previously Presented) The method of claim 98 wherein suspending the first thread of execution in response to the second opcode comprises:

testing whether the monitor event is pending;

testing whether a monitor is active;

if the monitor is active and no monitor event is pending, then entering a first thread suspended state.

107. (Previously Presented) The method of claim 106 wherein entering the first thread suspended state comprises:

relinquishing a plurality of registers in a register pool;

relinquishing a plurality of instruction queue entries in an instruction queue;

relinquishing a plurality of store buffer entries in a store buffer;

relinquishing a plurality of re-order buffer entries in a re-order buffer.

108. (Currently Amended) A system comprising:

a memory to store a loop ~~a first instruction~~ from a first thread, said loop including a first instruction, a second instruction and a test to determine whether data at a monitor address has changed and to restart said loop if said data at said monitor address remains unchanged, the first instruction having an associated address operand specified by an operand, the operand being an implicit operand in a predetermined register indicating said~~a~~ monitor address;

a first processor coupled to said memory, said first processor to enable a monitor to monitor memory transactions to detect a memory access to said monitor address in response to the first instruction and to cause resumption of said first thread in response to the memory access to the monitor address.

109. (Previously Presented) The system of claim 108 wherein said memory is to store a second instruction from said first thread, and wherein said first processor is to suspend said first thread in response to the second instruction.

110. (Previously Presented) The system of claim 109 wherein said monitor is to set a monitor event pending indicator in response the memory access occurring, said monitor event pending indicator to cause said first processor to resume a thread once unmasked by said second instruction.

111. (Previously Presented) The system of claim 108 wherein said first processor includes a first cache, the system further comprising:

a second processor comprising a second cache, wherein said first processor drives a bus transaction to the second processor to force said second processor to broadcast to the first processor any transactions that allow alteration of data stored at the monitor address in the second cache.

112. (Previously Presented) The system of claim 111 wherein said first processor is to assert a signal preventing said second processor from caching data at the monitor address in a state which would allow the second processor to modify data stored at the monitor address in the second cache without broadcasting that a modification is occurring.

113. (Previously Presented) The system of claim 112 wherein said signal indicates a cache hit and prevents the second cache from storing data at the monitor address in an exclusive state.

114. (Previously Presented) The system of claim 110 wherein said first processor is further to resume the first thread if an alternative event occurs.

115. (Previously Presented) The system of claim 114 wherein said alternative event is an interrupt.

116. (Cancelled).